UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/723,979	11/26/2003	David B. Gilgen	RSW920030248US1 (133)	9498

46320 7590 12/22/2009
CAREY, RODRIGUEZ, GREENBERG & PAUL, LLP
STEVEN M. GREENBERG
950 PENINSULA CORPORATE CIRCLE
SUITE 3020
BOCA RATON, FL 33487

EXAMINER	
DENG, ANNA CHEN	
ART UNIT	PAPER NUMBER
2191	
MAIL DATE	DELIVERY MODE
12/22/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte DAVID B. GILGEN
and WILLIAM D. WIGGER

Appeal 2009-002901
Application 10/723,979[1]
Technology Center 2100

Decided: December 22, 2009

Before LEE E. BARRETT, HOWARD B. BLANKENSHIP, and
JAMES R. HUGHES, *Administrative Patent Judges.*

BARRETT, *Administrative Patent Judge.*

DECISION ON APPEAL

This is a decision on appeal under 35 U.S.C. § 134(a) from the final rejection of claims 1-20. We have jurisdiction pursuant to 35 U.S.C. § 6(b).

We reverse.

---

[1] Filed November 26, 2003, titled "Fast Detection of the Origins of Memory Leaks When Using Pooled Resources."

STATEMENT OF THE CASE

The invention

The invention relates to detecting memory leaks and also the origin of memory leaks in connection with the allocation of resources from a resource pool to requesting code segments. Memory leakage is the gradual loss of allocable memory due to the failure to de-allocate previously allocated, but no longer utilized memory. Spec. ¶ [0003]. The resources in the resource pool can include any allocable process which can range from a communications socket to data processing logic. The requesting code segments, by comparison, can include any executable or interpretable logic such as a calling method in an object. A pool manager coupled to the resource pool is configured to allocate idle resources to requesting code segments and to determine the origin of memory leaks. Spec. ¶ [0016].

Calling code segments request the use of individual resources from the resource pool. The calling code segments can utilize the allocated resources, and subsequently, the calling code segments can return the allocated resources to the resource pool. Spec. ¶ [0017].

In operation, when a calling code segment requests the allocation of a resource from the resource pool, the pool manager locates the requested resource and appends a time stamp to the resource to indicate the time when the resource had been allocated. The pool manager passes the resource with time stamp to the calling code segment and retains a reference to the resource. During the allocation process, the pool manager inspects the call stack to identify the calling code segment. Finally, the pool manager writes a record to the data store of allocated resources containing the identity of both of the allocated resource and the calling code segment. Spec. ¶ [0020].

Periodically, the pool manager can validate each allocated resource to determine whether the resource allocations remain valid, or whether any one or more of the resource allocations have become overly idle. In this regard, the pool manager traverses the data store of allocated resources. For each allocated resource, the pool manager can query the resource to inspect the time stamp. When too much time has elapsed without the resource having been accessed by the calling code segment, the resource will have been presumed to have become overly and intolerably idle. As the calling code segment had not returned the resource to the resource pool, it further will be presumed that a memory leak has arisen. To facilitate remediation of the memory leak, a report of the memory leak detection can be generated along with the identity of the calling code segment. Spec. [0021].

Illustrative claim

Claim 1 is reproduced below for illustration:

> 1. A memory leak detection and reporting method comprising the steps of:
>
> time stamping allocated ones of resources in a resource pool;
>
> identifying calling code segments receiving said allocated resources;
>
> detecting memory leaks by inspecting individual timestamps for said allocated resources to determine whether said allocated resources have become overly idle; and,

for each allocated resource determined to have become overly idle, reporting an identity of a corresponding one of said calling code segments.

The references

Tarditi	US 6,625,808 B1	Sept. 23, 2003
Fu	US 2004/0172579 A1	Sept. 2, 2004 (filed Feb. 28, 2003)
Dahlstedt	US 2004/0133895 A1	Jul. 8, 2004 (Provisional application filed Dec. 20, 2002)

The rejections

Claims 7-9 stand rejected under 35 U.S.C. § 101 as being directed to nonstatutory subject matter.

Claims 1-3, 5-17, 19, and 20 stand rejected under 35 U.S.C. § 103(a) as unpatentable over Dahlstedt and Tarditi.

Claims 4 and 18 stand rejected under 35 U.S.C. § 103(a) as unpatentable over Dahlstedt and Tarditi, further in view of Fu.

35 U.S.C. § 101

Principle of law

It is recognized that subject matter must fall within one of the four statutory categories of 35 U.S.C. § 101 to be patent eligible. *See In re Nuijten*, 500 F.3d 1346, 1354 (Fed. Cir. 2007) ("If a claim covers material not found in any of the four statutory categories, that claim falls outside the plainly expressed scope of § 101 even if the subject matter is otherwise new and useful."). For example, a "signal" cannot be patentable subject matter because it is not within any of the four categories. *Id.* at 1357. Similarly, a "paradigm" does not fit within any of the four categories. *In re Ferguson*, 558 F.3d 1359, 1366 (Fed. Cir. 2009).

"Software" claimed, for example, as "executable code" or as mere functions is similarly not within any of the four statutory categories and is not patent eligible subject matter. *See In re Chatfield*, 545 F.2d 152, 159 (CCPA 1976) (Rich, J., dissenting) ("It has never been otherwise than perfectly clear to those desiring patent protection on inventions which are new and useful programs for general purpose computers (software) that the only way it could be obtained would be to describe and claim (35 U.S.C. § 112) the invention as a 'process' or a 'machine.'"). It is now also common to claim software as a "manufacture" by reciting storing program code stored on a tangible medium that, when executed, will perform a method. But, software per se is not a "manufacture" under § 101.

Issue

Is the subject matter of claims 7-9 software per se that falls outside the statutory categories of 35 U.S.C. § 101?

Analysis

The Examiner concludes that claims 7-9 are directed to nonstatutory subject matter because they recite computer listings per se, and the limitations are not physical "things." Final Rej. 2; Ans. 3.

The preamble of claim 7 recites a "memory leak detection and reporting system." A "system" is normally considered to be apparatus. *See*

In re Walter, 618 F.2d 758, 762 n.2 (CCPA 1980). However, we look to the body of the claim to see if it is consistent with the preamble.

Appellants argue that claim 7 is directed to hardware because it recites "a pool manager programmed . . ." (emphasis added) and that software alone is incapable of being "programmed." Br. 6. Appellants also argue that the claimed "data store" must be considered hardware since "[s]oftware alone is incapable of storing anything since it is functional descriptive material or an abstract idea." *Id.* It is argued that the Examiner has not explained how the claimed "data store" can store data without being connected to hardware. *Id.*

It is difficult to decide whether the limitation of "a pool manager programmed" is structure or just software. The term "pool manager" by itself could refer to a software program. The limitation of a "pool manager programmed" on one hand indicates that a structure is programmed because we agree with Appellants that software is not usually referred to as being programmed. On the other hand, there is no structure on a computer that corresponds to a pool manager--a pool manager is a software program. Nevertheless, because of the way the limitation is written as "a pool manager programmed," we interpret the limitation to be a programmed structure.

In any case, we agree with Appellants that the claimed "data store" must be interpreted as apparatus since software by itself is incapable of storing anything. The Examiner has not argued a claim interpretation that would suggest otherwise. Appellants have shown error in the conclusion that the claims are directed to software per se.

Conclusion

The subject matter of claims 7-9 is not software per se that falls outside the statutory categories of 35 U.S.C. § 101. The rejection of claims 7-9 is reversed.

35 U.S.C. § 103(a)

Principles of law

Obviousness requires that the differences between the subject sought to be patented and the prior art are such that the subject matter as a whole would have been obvious to one of ordinary skill in the art and that the combination teaches all claim limitations. 35 U.S.C. § 103(a).

Findings of fact

Dahlstedt

Dahlstedt describes detection of memory leaks in a garbage collected environment. Objects are stamped with a time stamp when they are created and memory is allocated. Every time the object is accessed, i.e., referenced by another object, the time stamp is updated. ¶ [0017]; Figures 1 and 4.

At a particular check time, all of the objects are analyzed to check their time stamps for the last access. Each object is categorized and marked as "warm" if the difference between the current time and the last access time is less than a certain specified time, or "cold" if the difference is greater than the specified time. The cold objects that point to each other are clustered together, and the warm objects that point to each other are clustered together. The warm object clusters that point to cold object clusters represent potential large memory leaks. ¶ [0022]; Figure 4. This information is provided to the software developer for use in detecting and analyzing potential memory leads. Abstract.

Dahlstedt teaches "time stamping allocated ones of resources in a resource pool" and "detecting memory leaks by inspecting individual timestamps for said allocated resources to determine whether said allocated resources have become overly idle" in claim 1, where "cold" objects correspond to "overly idle" resources.

Dahlstedt does not teach or suggest "identifying calling code segments receiving said allocated resources" or "for each allocated resource determined to have become overly idle, reporting an identity of a corresponding one of said calling code segments."

Tarditi

Tarditi describes a system and method for facilitating memory management among heterogeneous elements of a program. Heterogeneous programs comprise two different kinds of programming language, e.g., an advanced programming language supporting automated garbage collection and a legacy programming language that does not support automated garbage collection. It is often more convenient to utilize code written in legacy programming languages than to attempt to re-write the code using an advanced language. Col. 2, ll. 44-48. Recognizing this need, nearly every runtime environment for an advanced programming language now provides a foreign function interface to support calls to "foreign functions" written in legacy code. Col. 2, ll. 52-56.

The problem is that automated memory management functions are suspended when a call is made to the foreign function. Insofar as the foreign function employs data structures that are different from, yet occupy the same address space as, the data structures of the advanced programming language, it is probable that the manual memory management of the foreign language function will corrupt pointers to objects stored within the heap, especially if the objects are utilized or referenced by the foreign function. Since the garbage collection function relies on such pointers to identify live objects on the heap, corruption of the pointers retards the garbage collectors ability to reclaim memory from dead objects and may eliminate a reference to a live object. Col. 3, ll. 4-18.

Tarditi's solution provides an analysis agent which detects calls to foreign functions and returns from foreign functions within the source code of a high-level programming language. Once identified, the analysis agent generates and inserts the code necessary to implement a transition function in the program flow for that thread. More specifically, the analysis agent inserts a creation function at a transition point from the advanced programming language having automated garbage collection to a foreign function not having automated garbage collection and inserts a restoration function at a transition point from a foreign function back to the advanced programming language. Col. 9, ll. 8-22. The transition function (creation and restoration function) maintains a history of select pointer and state information throughout program execution, which prevents the loss of pointer information necessary for garbage collection to accurately identify live objects. Col. 4, ll. 48-56.

As shown in Figure 4, a call stack 402 for Thread 1 is broken into "chunks" or frames to indicate program components. The frames are designated as either a GC frame or a non-GC frame. GC denotes that the frame was developed in an advanced programming language that supports garbage collection. Col. 11, ll. 18-27. Transition functions comprising paired creation functions 304 and transition functions 306 have been integrated into the program flow. Col. 11, ll. 28-32. For each transition from a GC frame to a non-GC frame in the call stack, creation function 304 allocates space on the stack frame for a transition record storing select pointer and state information as well as a pointer to the immediate past transition record. Col. 11, ll. 42-48.

Contentions

The Examiner finds that Dahlstedt does not teach "identifying calling code segments receiving said allocated resources" or "for each allocated resource determined to have become overly idle, reporting an identity of a corresponding one of said calling code segments." The Examiner finds that Tarditi teaches these limitations at column 11, lines 10-49, which describes integrating a creation function on the call stack which allocates space on the stack frame for a transition record. The Examiner concludes that it would have been obvious to modify Dahlstedt to include "identifying calling code segments receiving said allocated resources" and "for each allocated resource determined to have become overly idle, reporting an identity of a corresponding one of said calling code segments" as claimed "because one of ordinary skill in the art would be motivated to facilitate automated memory management among heterogeneous components of a computer program (Tarditi, col. 4, lines 40-42)." Final Rej. 3-4; Ans. 4-5.

Appellants argue that the Examiner's analysis fails to consider the claimed invention as a whole. Br. 9. Claim 1 recites: "*for each allocated resource determined to have become overly idle*, <u>reporting an identity of a corresponding one of said calling code segments</u>." (Emphasis added.) It is argued that the second clause (the underlined clause) is conditioned upon the first clause (the italicized clause), but the Examiner relies on Dahlstedt to teach the limitations of the first clause and Tarditi to teach the limitations of the second clause without any rationale for linking the references. Br. 9-10. It is argued that the Examiner's rationale is based on Appellants' own disclosure and constitutes impermissible hindsight. *Id.* at 10.

Appellants argue that the Examiner's rationale for the proposed modification, "to facilitate automated memory management among heterogeneous components of a computer program," is no more than a generalization and does not establish that one of ordinary skill would have been "<u>realistically impelled</u> to modify the prior art in the manner suggested by the Examiner." Br. 11

The Examiner responds that the transition records in Tarditi read on the limitation of "reporting an identity of a corresponding one of said calling code segments" and thus the combination of Dahlstedt and Tarditi teach all of the limitations of claim 1. Ans. 14.

Appellants do not dispute that Dahlstedt and Tarditi teach the limitations at issue individually, and that "Tarditi could be considered as teaching identifying call code segments receiving allocated resources." Reply Br. 4. However, Appellants argue, the Examiner's failure to consider the claim as a whole "is evidenced by the Examiner's failure to establish that the prior art teaches the limitations 'for each' and 'a corresponding one of' found in the limitation at issue." *Id.* It is argued that the combination of references does not teach that "a corresponding one of" the code segments is reported and do not associate "each" of the objects with corresponding calling code segments when the objects are considered cold. *Id.* Appellants argue that the only way one of ordinary skill in the art could have arrived at the limitations "for each" and "a corresponding one of" is by the using Appellants' disclosure. *Id.* at 6.

Issue

Appellants do not dispute that Dahlstedt teaches "time stamping allocated ones of resources in a resource pool" and "detecting memory leaks by inspecting individual timestamps for said allocated resources to determine whether said allocated resources have become overly idle." Appellants acknowledge that Tarditi could be considered as "identifying calling code segments receiving said allocated resources." Reply Br. 4. Therefore, the issue in dispute is:

Have Appellants demonstrated that the Examiner erred in concluding that the combination of Dahlstedt and Tarditi would have made the limitation of "for each allocated resource determined to have become overly idle, reporting an identity of a corresponding one of said calling code segments" in claim 1 obvious to one of ordinary skill in the art?

Analysis

We do not see how the transition records in Tarditi correspond to "identifying calling code segments receiving said allocated resources" and "reporting an identity of a corresponding one of said calling code segments" in claim 1 as found by the Examiner. Transition records store select pointer and state information as well as a pointer to the immediate past transition record, but the Examiner has not explained how this information identifies a code segment receiving an allocated resource. For example, GC frame code segment 408 in Figure 4 has allocated resources for Object 1 on the heap, but it does not have a transition record because it is immediately followed by another GC frame code segment. Furthermore, non-GC frame code segments do not have transition records. It appears that the Examiner may have assumed that because Tarditi refers to a "call stack" in Figure 4 that it operates in the same way as Appellants' invention which is described as having a "call stack." The mere use of common language is not enough to show identity of function. Nevertheless, since Appellants acknowledge that Tarditi could be considered as "identifying calling code segments receiving said allocated resources," we assume that it is taught for this appeal.

The limitation at issue is: "*for each allocated resource determined to have become overly idle*, <u>reporting an identity of a corresponding one of said calling code segments</u>." We agree with Appellants that the second clause (the underlined clause) is conditioned upon the first clause (the italicized clause), and thus it must be shown that it would have been obvious to report an identity of a calling code segment for each allocated resource determined to have become overly idle, not just reporting the identity of calling code segments. We agree with Appellants that the statement of the rejection does not address this interpretation, but treats the two clauses independently.

Dahlstedt teaches the italicized limitation of determining allocated resource which have become overly idle (i.e., cold), but does not teach or suggest that the resource determined to be overly idle is identified with a calling code segment. Assuming, *arguendo*, that Tarditi teaches "identifying calling code segments receiving said allocated resources," it might be assumed that Tarditi teaches "reporting an identity of . . . calling code segments." However, as Appellants argue, the limitation requires "reporting an identity of <u>a corresponding one of</u> said calling code segments" which refers back to "<u>each</u> allocated resource determined to have become overly idle." Tarditi does not identify resources that have become idle for too long. Tarditi supports the garbage collection function of the runtime environment of the advanced programming language. Col. 9, ll. 4-7. Garbage collection is done by the conventional process of "walking the stack" to identify all live objects and return "dead" objects to the heap. Col. 2, ll. 9-27; col. 10, ll. 42-58. A "dead" object is not the same as an "overly idle" resource determined by a time stamp. Even if the "dead" object could be considered analogous to the claimed "overly idle" resource, Tarditi does not associate "dead" objects with a calling code segment and thus does not suggest the limitation of "for each allocated resource determined to have become overly idle, reporting an identity of a corresponding one of said calling code segments." The Examiner's stated motivation, "to facilitate automated memory management around heterogeneous components of a computer system" (Ans. 5), would only be persuasive if the two clauses at issue were separate and independent, which they are not. Accordingly, we conclude that the combination of the teachings of Dahlstedt and Tarditi could not have made the subject matter of claim 1 obvious to one of ordinary skill in the art.

Conclusion

Appellants have demonstrated that the Examiner erred in concluding that the combination of Dahlstedt and Tarditi would have made the limitation of "for each allocated resource determined to have become overly idle, reporting an identity of a corresponding one of said calling code segments" in claim 1 obvious to one of ordinary skill in the art. The rejection of claim 1 is reversed. Independent claims 7, 10, 15, and 20 contain corresponding limitations. Accordingly, the rejection of claims 1-3, 5-17, 19, and 20 is reversed. Fu, which is applied to the rejection of claims 4 and 18, is not applied for the limitations missing in Dahlstedt and Tarditi. Accordingly, the rejection of claims 4 and 18 is also reversed.

CONCLUSION

The rejection of claims 7-9 under 35 U.S.C. § 101 is reversed.

The rejections of claims 1-20 under 35 U.S.C. § 103(a) are reversed.

<u>REVERSED</u>

erc

CAREY, RODRIGUEZ, GREENBERG & PAUL, LLP
STEVEN M. GREENBERG
950 PENINSULA CORPORATE CIRCLE
SUITE 3020
BOCA RATON, FL 33487